Министерство образования республики Молдова

Технический Университет Молдовы

Департамент Программной Инженерии и Автоматики

Отчёт

Лабораторная работа №3

По предмету: Programarea in retea

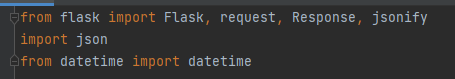
Выполнил студ. гр. SI-202 Абабий Эдуард

Проверил Лях Аркадий

Кишинёв – 2023

Тема: Comunicarea în Web: Protocol și aplicație client HTTP

Для начала создал файл server.py, ( локальный сервер ), который будет обрабатывать запросы GET POST HEAD OPTIONS



Импортируем все необходимые зависимости, для создания сервера использую инструмент flask

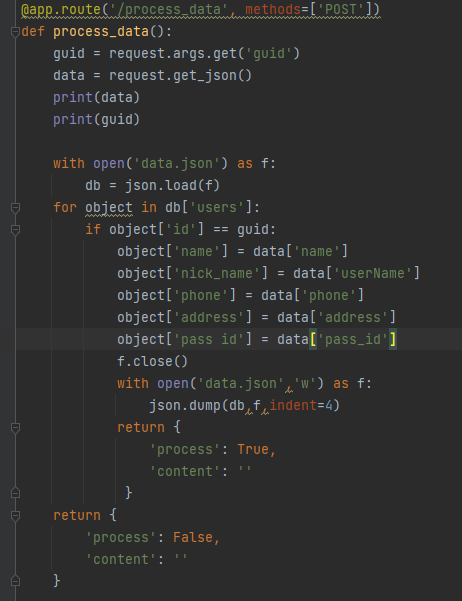


В данной функции определен обработчик маршрута /get\_data\_guid, который принимает запросы только методом GET. Ожидается, что в параметрах запроса будет передан параметр guid.

При обработке запроса, сначала получается значение параметра guid через объект request.args. Затем открывается файл data.json и загружается в объект data.

Далее, выполняется цикл по всем объектам, содержащимся в массиве users из объекта data. Если находится объект с id, соответствующим переданному параметру guid, то он возвращается в виде JSON-объекта с помощью функции return. Если ни один объект не соответствует переданному guid, то возвращается JSON-объект с ключом process, равным False, и пустым значением в ключе content.

Таким образом, данная функция ищет объект с заданным guid в массиве объектов users из data.json и возвращает его в JSON-формате, если он найден. Если же объект не найден, возвращается JSON-объект с флагом process, указывающим на неудачу, и пустым содержимым.



Это обработчик POST-запроса на адрес '/process\_data'. При получении такого запроса сервер должен прочитать параметр 'guid' из URL-адреса, данные в формате JSON из тела запроса, и затем обновить запись о пользователе с соответствующим идентификатором (guid) в базе данных, используя данные из тела запроса.

Затем происходит следующее:

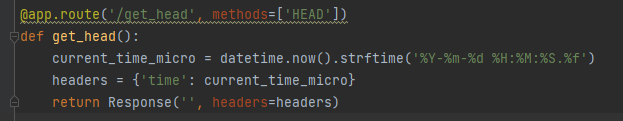
Сначала читается файл 'data.json' с помощью оператора 'with open() as f', чтобы получить доступ к данным в файле.

Затем данные в файле загружаются в переменную db с помощью функции json.load().

После этого сервер перебирает все объекты в массиве 'users' и, если находит объект с идентификатором, соответствующим guid, обновляет его поля с помощью данных из тела запроса.

После этого файл закрывается с помощью f.close() и данные сохраняются в файл с помощью json.dump() в том же формате, что и изначально (с отступами indent=4).

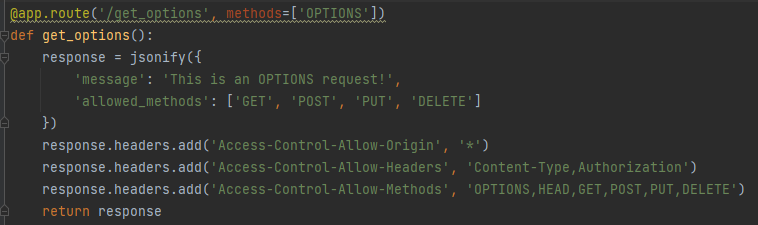
В конце функция возвращает JSON-объект, который сообщает клиенту о том, прошло ли обновление успешно. Если обновление было успешно, значение 'process' равно True, иначе False.



Данный код является обработчиком маршрута для HTTP-метода HEAD в рамках веб-приложения Flask. HTTP-метод HEAD используется для получения только заголовков ответа от сервера без тела ответа.

При обращении к данному маршруту методом HEAD сервер возвращает пустой ответ, но в заголовки ответа добавляется пользовательский заголовок 'time', который содержит текущее время в микросекундах в определенном формате.

Таким образом, когда пользователь обратится к данному маршруту методом HEAD, сервер вернет пустой ответ с добавленным пользовательским заголовком 'time', который содержит текущее время в микросекундах в определенном формате.



Когда пользователь обращается к данному маршруту методом OPTIONS, сервер возвращает JSON-объект в ответе, содержащий сообщение 'This is an OPTIONS request!' и список доступных методов ('GET', 'POST', 'PUT', 'DELETE').

Для того, чтобы клиентский код мог обращаться к данному маршруту с других доменов, сервер добавляет в заголовки ответа следующие поля:

'Access-Control-Allow-Origin': '\*' - разрешает доступ к ресурсу со всех доменов

'Access-Control-Allow-Headers': 'Content-Type,Authorization' - разрешает использование заголовков 'Content-Type' и 'Authorization'

'Access-Control-Allow-Methods': 'OPTIONS,HEAD,GET,POST,PUT,DELETE' - разрешает использование HTTP-методов 'OPTIONS', 'HEAD', 'GET', 'POST', 'PUT' и 'DELETE'

Таким образом, при обращении к данному маршруту методом OPTIONS, сервер вернет JSON-объект с информацией о доступных методах и свойствах ресурса, а также добавит в заголовки ответа разрешения для обращения к ресурсу из других доменов.



Данный код представляет собой JSON-объект, содержащий список пользователей с их информацией.

Каждый пользователь представлен в объекте в виде словаря, который содержит следующие поля:

"id": уникальный идентификатор пользователя

"email": адрес электронной почты пользователя

"nick\_name": никнейм пользователя

"name": имя пользователя

"phone": номер телефона пользователя

"address": адрес пользователя

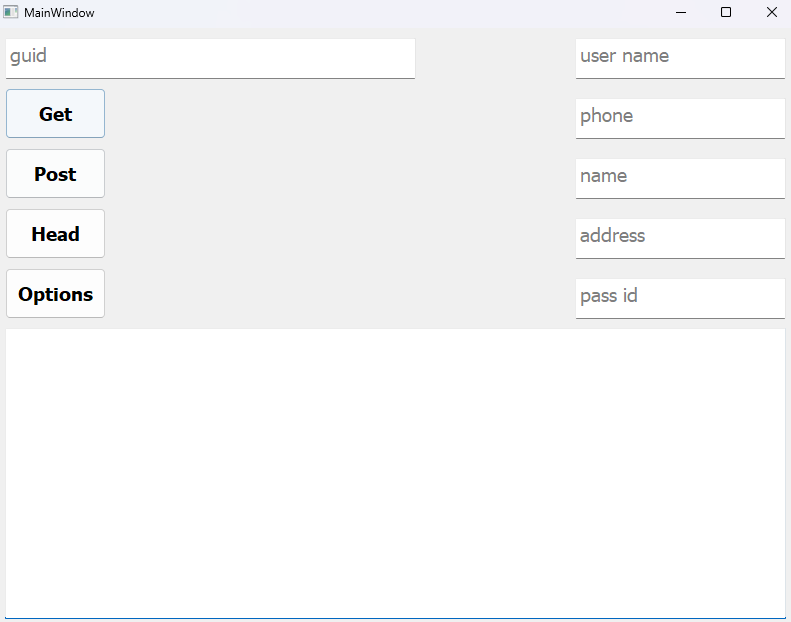
"pass id": идентификационный номер паспорта пользователя

Объект содержит список из трех пользователей. Каждый пользователь имеет уникальный идентификатор "id". Каждое поле содержит соответствующую информацию о пользователе.

Данный объект может использоваться для передачи информации о пользователях между приложениями или компонентами программного обеспечения. Он может быть обработан и представлен в удобном виде с помощью кода на разных языках программирования, например, на Python, JavaScript, Java и др.

Я его создал, как псевдо базу данных, которая будет обрабатываться во время запросов

Затем я взялся за создание небольшого GUI приложения



У нас есть несколько полей ввода с placeholder.

Guid – поле ввода для идентификатор пользователя

User name – поле для ввода ника пользователя ( для дальнейшей смены )

Phone - поле для ввода телефона пользователя ( для дальнейшей смены )

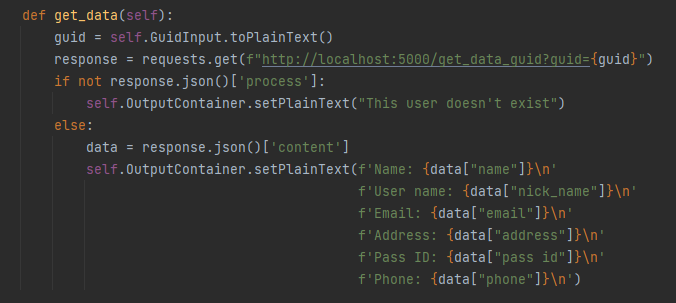
Name - поле для ввода имени пользователя ( для дальнейшей смены )

Address – поле для ввода адреса пользователя ( для дальнейшей смены )

Pass id – поле для ввода паспортных данных ( для дальнейшей смены )

Большое поле, которое пустое, оно нам понадобится для вывода результата запросов

Думаю объяснять смысл кнопок не нужно, каждая создаёт определенный запрос



Данный код является методом класса и представляет собой клиентскую часть для получения данных о пользователе из удаленного сервера.

Метод принимает GUID пользователя в виде строки, которая извлекается из виджета GuidInput. Затем метод отправляет GET-запрос на сервер с параметром guid, содержащим переданный GUID.

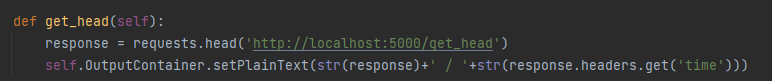
Если на сервере не найден пользователь с переданным GUID, метод выводит сообщение об этом в виджет Output Container. В противном случае метод получает данные о пользователе в формате JSON из поля "content" в ответе сервера и выводит информацию о пользователе в виджет OutputContainer в виде строки с помощью f-строки, содержащей соответствующие поля: имя, никнейм, адрес, идентификационный номер паспорта, электронная почта и телефон.



Данный код является методом класса и представляет собой клиентскую часть для отправки данных о пользователе на удаленный сервер.

Метод принимает GUID пользователя в виде строки, который извлекается из виджета GuidInput, а также данные о пользователе, которые заполняются в соответствующих виджетах NameInput, UserNameInput, AddressInput, PhoneInput и PassInput. Далее данные пользователя упаковываются в формат JSON и отправляются на сервер методом POST с указанием GUID в параметре запроса.

Если на сервере не найден пользователь с переданным GUID, метод выводит сообщение об этом в виджет Output Container. В противном случае метод выводит сообщение об успешной отправке данных в виджет OutputContainer.

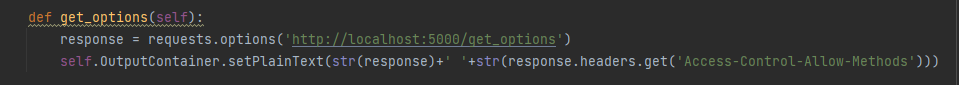


Данный метод отправляет HTTP-запрос методом HEAD на адрес http://localhost:5000/get\_head.

В ответ на этот запрос сервер должен вернуть только заголовки ответа без тела.

Затем, метод получает ответ от сервера, извлекает значение заголовка time и выводит его в Output Container.

Заголовок time был добавлен на сервере в ответ на запрос методом HEAD, вернув пустое тело ответа, чтобы клиент мог узнать текущее время на сервере.



Этот метод отправляет HTTP-запрос методом OPTIONS на адрес http://localhost:5000/get\_options.

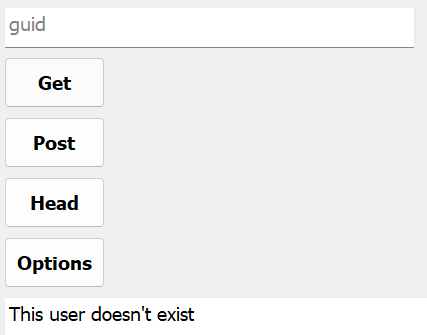
Метод OPTIONS используется для запроса информации о доступных методах для определенного ресурса на сервере.

В ответ на этот запрос сервер должен вернуть список разрешенных методов и другую информацию.

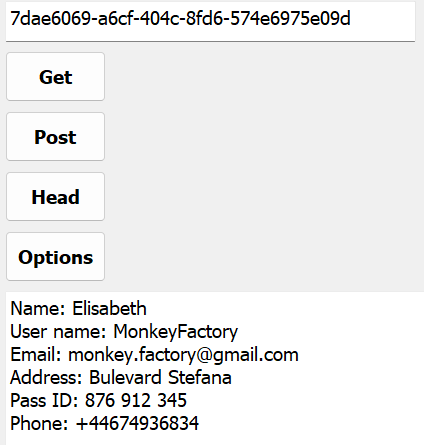
Затем, метод получает ответ от сервера, извлекает значение заголовка Access-Control-Allow-Methods и выводит его в OutputContainer.

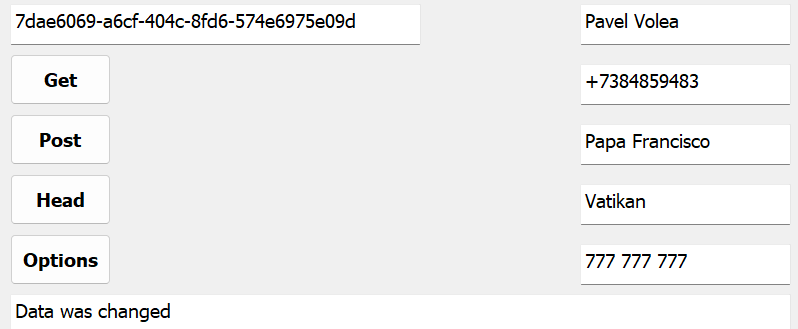
Этот заголовок был добавлен на сервере в ответ на запрос методом OPTIONS, чтобы указать, какие методы HTTP разрешены для данного ресурса и могут быть использованы клиентом.

Результат работы запросов:

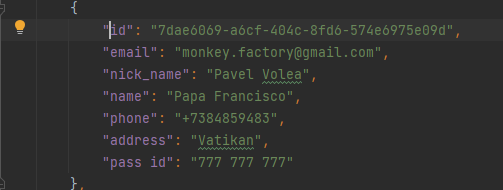
Обработка запроса GET без указания GUID пользователя

С указанием GUID пользователя

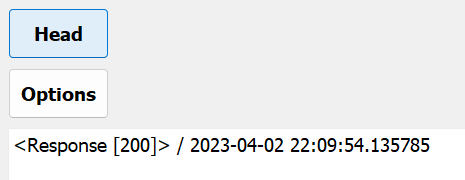


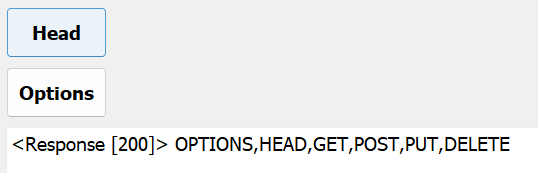


Обработка запроса POST с данными   
Результат запроса



Результат запроса HEAD



Результат запроса OPTIONS

Вывод: В данной лабораторной работе из нового для себя я узнал инструмент flask для создания локального сервера обработчика запросов, но с ним ещё предстоит познакомиться получше. Создал небольшое GUI приложение для обработки запросов GET POST HEAD OPTIONS